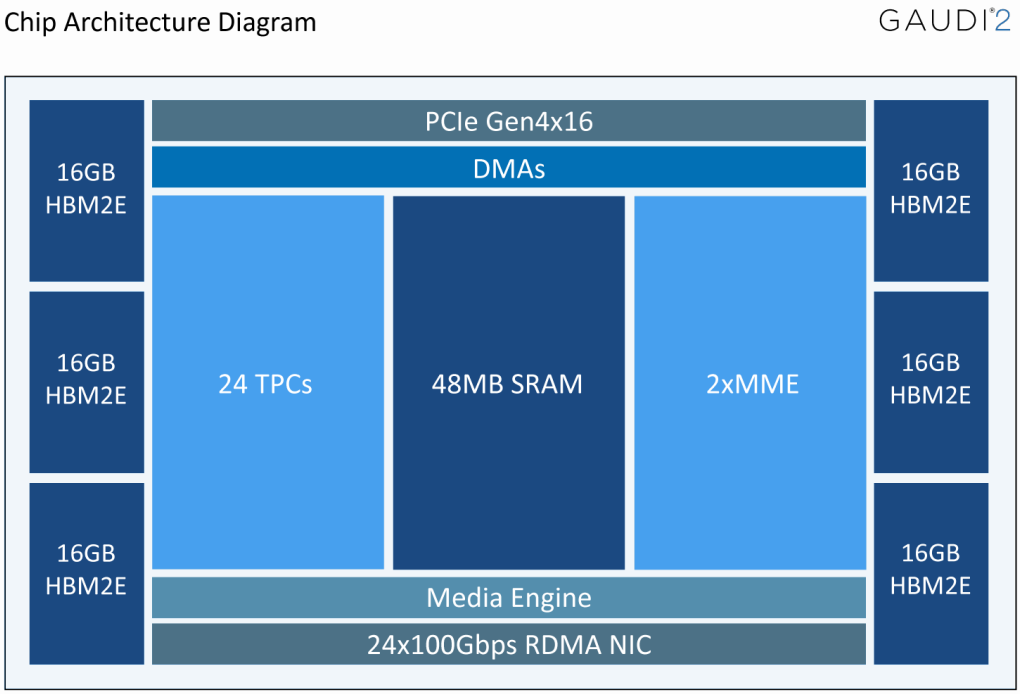
Intel is sitting on a huge amount of card inventory they can't move, largely because of bad software. Most of this is a summary of the public #intel-hardware channel in the tinygrad discord.

Intel currently is sitting on:

- 15,000 Gaudi 2 cards (with baseboards)
- 5,100 Intel Data Center GPU Max 1450s (without baseboards)

If you were Intel, what would you do with them?

First, starting with the Gaudi cards. The open source repo needed to control them was archived on Feb 4, 2025. There's a closed source version of this that's *maybe* still maintained, but eww closed source and do you think it's really maintained?



The architecture is kind of tragic, and that's likely why they didn't open source it. Unlike every other accelerator I have seen, the MMEs, which is where all the FLOPS are, are not controllable by the TPCs. While the TPCs have an LLVM port, the MME is not documented. After some poking around, I found the spec:

```
typedef struct _Desc
{
    MmeAddress baseAddrCOut1;                       // address of the second COut operand
    MmeAddress baseAddrCOut0;                       // address of the first COut operand
    MmeAddress baseAddrA;                           // address of operand A
    MmeAddress baseAddrB;                           // address of operand B
    MmeBrainsCtrl brains;                            // Brain CTRL info.
    MmeHeader header;                                // The operation header.
    MmeCtrl ctrl;                                    // Routing and EU sync info.
    MmeTensorDesc tensorA;                           // The tensor of operand A.
    MmeTensorDesc tensorB;                           // The tensor of operand B.
    MmeTensorDesc tensorCOut;                        // The tensor of operand COut.
    MmeSyncObject syncObject;                        // The sync object value and address.
    MmeAguCoreDesc aguIn[c_mme_sb_nr][MME_CORE_PAIR_SIZE];          // The Input AGUs info
    uint32_t spatialSizeMinus1A;                     // spatial size for A
    uint32_t spatialSizeMinus1B;                     // spatial size for B slave and master.
    MmeAguCoreDesc aguOut[c_mme_wb_nr][MME_CORE_PAIR_SIZE];        // The Output AGUs info
    uint32_t spatialSizeMinus1Cout;                  // spatial size for C out.
    MmeConvDesc conv;                                // The convolution descriptor.
    MmeOuterLoop outerLoop;                          // Number of tetrises loops.
    uint32_t numIterationsMinus1;                    // The number of consecutive activations (number of tetrises).
    MmeSBRepeat sbRepeat;                            // SB rewind info.
    MmeFP8Bias fp8Bias;                              // bias values for fp8 1-5-2
    MmeRateLimiter rateLimiter;                      // RL info.
    MmeUserData axiUserData;                         // AXI user data.
    MmePerfEvt perfEvtIn;                            // Performance event for input operands.
    MmePerfEvt perfEvtOut;                           // Performance event for output operands.
    MmePCU pcu;                                      // PCU RL info.
    uint32_t slaveSyncObject0Addr;                   // Slave SO0 address
    uint32_t slaveSyncObject1Addr;                   // Slave SO1 address
    MmePowerLoop powerLoop;                          // Power Loop info
    MmeSpare spare[MME_CORE_PAIR_SIZE];              // Spare bits.
    uint32_t wkldID;                                 // workload ID
} Desc;
```

It's highly fixed function, looks very similar to the Apple ANE. But that's not even the real problem with it. The problem is that it is controlled by queues, not by the TPCs. Unpacking `habanalabs-dkms-1.19.2-32.all.deb` you can find the queues.

```
enum gaudi2_queue_id {
    GAUDI2_QUEUE_ID_PDMA_0_0 = 0,
    GAUDI2_QUEUE_ID_PDMA_0_1 = 1,
    GAUDI2_QUEUE_ID_PDMA_0_2 = 2,
    GAUDI2_QUEUE_ID_PDMA_0_3 = 3,
    GAUDI2_QUEUE_ID_PDMA_1_0 = 4,
    GAUDI2_QUEUE_ID_PDMA_1_1 = 5,
    GAUDI2_QUEUE_ID_PDMA_1_2 = 6,
    GAUDI2_QUEUE_ID_PDMA_1_3 = 7,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_0_0 = 8,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_0_1 = 9,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_0_2 = 10,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_0_3 = 11,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_1_0 = 12,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_1_1 = 13,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_1_2 = 14,
    GAUDI2_QUEUE_ID_DCORE0_EDMA_1_3 = 15,
    GAUDI2_QUEUE_ID_DCORE0_MME_0_0 = 16,
    GAUDI2_QUEUE_ID_DCORE0_MME_0_1 = 17,
    GAUDI2_QUEUE_ID_DCORE0_MME_0_2 = 18,
    GAUDI2_QUEUE_ID_DCORE0_MME_0_3 = 19,
    GAUDI2_QUEUE_ID_DCORE0_TPC_0_0 = 20,
    GAUDI2_QUEUE_ID_DCORE0_TPC_0_1 = 21,
    GAUDI2_QUEUE_ID_DCORE0_TPC_0_2 = 22,
```

There is some way to push a command stream to the device so you don't actually have to deal with the host itself for the queues. But that doesn't prevent you having to decompose the network you are trying to run into something you can put on this fixed function block.

Programmability is on a spectrum, ranging from CPUs being the easiest, to GPUs, to things like the Qualcomm DSP / Google TPU (where at least you drive the MME from the program), to this and the Apple ANE being the hardest. While it's impressive that they actually got on MLPerf Training v4.0 training GPT3, I suspect it's all hand coded, and if you even can deviate off the trodden path you'll get almost no perf.

Accelerators like this are okay for low power inference where you can adjust the model architecture for the target, Apple does a great job of this. But this will never be acceptable for a training chip.

Then there's the Data Center GPU Max 1450. Intel actually sent us a few of these, so good guy Intel thank you for the samples.

However, you quickly run into a problem…how do you plug them in? They need OAM sockets, 48V power, and a cooling solution that can sink 600W. As far as I can tell, they were only ever deployed in two systems, the Aurora

Supercomputer and the Dell XE9640. It's hard to know, but I doubt many of these Dell systems were sold.

Intel then sent us this carrier board. In some ways it's helpful, but in other ways it's not at all. It still doesn't solve cooling or power, and you need to buy 16x MCIO cables (cheap in quantity, but expensive and hard to find off the shelf). Also, I never got a straight answer, but I doubt Intel has many of these boards. And that board doesn't look cheap to manufacture more of. The connectors alone, which you need two of per GPU, cost $26 each. That's $208 for just the OAM connectors.

tiny corp was in discussions to buy these GPUs. How much would you pay for one of these on a PCIe card? The specs look great. 839 TFLOPS, 128 GB of ram, 3.3 TB/s of bandwidth. However…read this article. Even in simple synthetic benchmarks, the chip doesn't get anywhere near its max performance, and it looks to be for fundamental reasons like memory latency.

We estimate we could sell PCIe versions of these GPUs for $1,000; I don't think most people know how hard it is to move non NVIDIA hardware. Before you say you'd pay more, ask yourself, do you really want to deal with the software?

An adapter card has four pieces. A PCB for the card, a 12->48V voltage converter, a heatsink, and a fan. My quote from the guy who makes an OAM adapter board was $310 for the PCB in 10+ quantity and $75 for the voltage converter. A heatsink that can handle 600W (heat pipes + vapor chamber) is going to cost $100, then maybe $20 more for the fan. That's $505, and you still need to assemble and test them, oh and now there's tariffs. Maybe you can get this down to $400 in ~1000 unit quantity.

So $200 for the GPU, $400 for the adapter, $100 for shipping/fulfillment/returns (more if you use Amazon), and 30% profit if you sell at $1k. tiny would net $1M on this, which has to cover NRE and you have risk of unsold inventory. We offered Intel $200 per GPU (a $680k wire) and they said no. They wanted $600, negotiations were stupidly slow, and even at that price it seemed like there were going to be weird strings attached. I suspect that unless a supercomputer person who already uses these GPUs wants to buy more they will ride it to zero.

If I were Intel, tiny corp is exactly who I'd want to have these GPUs. Imagine I'm $680k in the hole on Intel hardware. I'm really incentivized to make the software good to move these units, and I suspect many of the improvements would translate to A770/B580. It would get the GPUs out into people's hands and get *someone* excited about this. But nope, I wasn't able to find an individual with the power to make a deal, never mind actually make a deal. It's unclear any individual in the company has that power. It makes me super reluctant to put any engineering effort into these GPUs.

tl;dr: there's 5100 of these GPUs with no simple way to plug them in. It's unclear if they worth the cost of the slot they go in, at least at normal slot prices. I bet they end up shredded, or *maybe* dumped on eBay for $50 each in a year like the Xeon Phi cards. If you buy one, good luck plugging it in!

The reason Meta and friends buy some AMD is as a hedge against NVIDIA. Even if it's not usable, AMD has progressed on a solid steady roadmap, with a clear continuation from the 2018 MI50 (which you can now buy for 99% off), to the MI325X which is a super exciting chip (AMD is king of chiplets). They are even showing signs of finally investing in software, which makes me bullish. If NVIDIA stumbles for a generation, this is AMD's game. The ROCm "copy each NVIDIA repo" strategy actually works if your competition stumbles. They can win GPUs with slow and steady improvement + competition stumbling, that's how AMD won server CPUs.

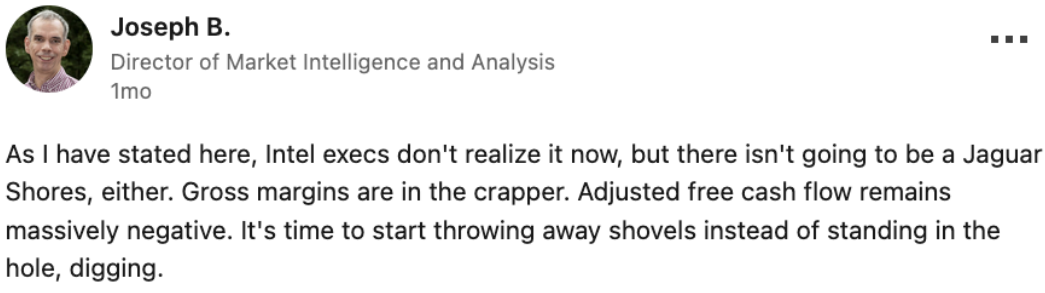
With these Intel chips, I'm not sure what companies they would appeal to. Ponte Vecchio is cancelled. There's no point in investing in the platform if there's not going to be a next generation, and therefore nobody can justify the cost of developing software, therefore there won't be software, therefore they aren't worth plugging in.

Where does this leave Intel's AI roadmap? The successor to Ponte Vecchio was Rialto Bridge, but that was cancelled. The successor to that was Falcon Shores, but that was also cancelled. Intel claims the next GPU will be "Jaguar Shores", but fool me once… To quote JazzLord1234 from reddit

> *"No point even bothering to listen to their roadmaps anymore. They have squandered all their credibility."*

Gaudi 3 is a flop due to "unbaked software", but as much as I usually do blame software, nothing has changed from Gaudi 2 and it's just a really hard chip to program for. So there's no future there either.

I can't say that "Jaguar Shores" square instills confidence. It didn't inspire confidence for "Joseph B." on LinkedIn either.



From my interactions with Intel people, it seems there's no individuals with power there, it's all committee like leadership. The problem with this is there's nobody who can say yes, just many people who can say no. Hence all the cancellations and the nonsense strategy.

AMD's dysfunction is different. from the beginning they had leadership that can do things (Lisa Su replied to my first e-mail), they just didn't see the value in investing in software until recently. They sort of had a point if they were only targeting hyperscalars. but it seems like SemiAnalysis got through to them that hyperscalars aren't going to deal with bad software either. It

remains to be seen if they can shift culture to actually deliver good software, but there's movement in that direction, and if they succeed AMD is so undervalued. Their hardware is good.

With Intel, until that committee style leadership is gone, there's 0 chance for success. Committee leadership is fine if you are trying to maintain, but Intel's AI situation is even more hopeless than AMD's, and you'd need something major to turn it around. At least with AMD, you can try installing ROCm from the PyTorch homepage and be frustrated when there are bugs. Every time I have managed even to find which piece of Intel software I was supposed to use, I can't recall even getting the import to work without a segfault or missing library.

Intel needs actual leadership to turn this around, or there's 0 future in Intel AI.